AMENDMENTS TO THE CLAIMS

1. (Original) A method for accessing a unified memory in a micro-processing system having a microprocessor, a one level pipeline, and a two-phase clock, such that all instructions are executed in a single cycle, comprising:

(a) fetching a program instruction from the unified memory;

(b) determining if the fetched program instruction would require three unified memory accesses during a single instruction cycle for proper execution of the fetched program instruction, proper execution of the fetched program instruction being the microprocessor performing the operations requested by the fetched program instruction in a single instruction cycle;

(c) accessing the unified memory a first time, during the instruction cycle associated with the fetched program instruction, with a dummy access when it is determined that the fetched program instruction requires three unified memory accesses for proper execution of the fetched program instruction;

(d) fetching a next program instruction from an instruction register, during the instruction cycle associated with the fetched program instruction, when it is determined that the fetched program instruction requires three unified memory accesses for proper execution of the fetched program instruction; and

(e) accessing the unified memory a second time, during the instruction cycle associated with the fetched program instruction, with a data access when it is determined that the fetched program instruction requires three unified memory accesses for proper execution of the fetched program instruction.

2. (Original) The method as claimed in claim 1, wherein the data access is a read data

access.

3. (Original) The method as claimed in claim 1, wherein the data access is a write data access.

4. (Original) The method as claimed in claim 1, wherein the fetched program instruction is a last instruction of a loop.

5. (Original) The method as claimed in claim 1, wherein the instruction register is an instruction stack, thereby enabling program instruction fetches for nested loops.

6. (Original) A method for accessing a unified memory in a micro-processing system having a microprocessor, a one level pipeline, and a two-phase clock, such that all instructions are executed in a single cycle, comprising:

(a) fetching a program instruction from the unified memory during a first instruction cycle;

(b) determining if the fetched program instruction for a second instruction cycle is a conditional program code discontinuity;

(c) accessing the unified memory a first time during the second instruction cycle with a dummy access when it is determined that the program instruction accessed for a second instruction cycle is a conditional program code discontinuity; and

(d) accessing the unified memory a second time during the second instruction cycle to read a new instruction when it is determined the program instruction accessed for a second instruction cycle is a conditional program code discontinuity, thereby delaying the instruction access from the unified memory for the second instruction cycle by a half cycle.

7. (Original) The method as claimed in claim 6, wherein the conditional program code

discontinuity is a jump instruction.

8. (Original) The method as claimed in claim 6, wherein the conditional program code discontinuity is a call instruction.

9. (Original) A method for accessing a unified memory in a micro-processing system having a microprocessor, a one level pipeline, and a two-phase clock, such that all instructions are executed in a single cycle, comprising:

(a) fetching a program instruction from the unified memory;

(b) determining if the fetched program instruction is a loop initiation instruction;

(c) storing a first instruction of the loop in an instruction register when the fetched program instruction is a loop initiation instruction;

(d) executing the loop;

(e) determining if a fetched instruction during the execution of the loop is a last instruction of the loop;

(f) accessing the unified memory a first time, during the instruction cycle associated with the fetched last instruction of loop, with a dummy access;

(d) fetching the first instruction of the loop from the instruction register, during the instruction cycle associated with the fetched last instruction of loop; and

(e) accessing the unified memory a second time, during the instruction cycle associated with the fetched last instruction of loop, with a data access.

10. (Original) The method as claimed in claim 9, wherein the data access is a read data access.

[12] 11. (Currently Amended) The method as claimed in claim 9, wherein the data access is a write data access.

[13] 12. (Currently Amended) The method as claimed in claim 9, wherein the instruction register is an instruction stack, thereby enabling program instruction fetches for nested loops.

[14] 13. (Currently Amended) A method for accessing a unified memory in a micro-processing system during a loop instruction, comprising:

(a) accessing a program instruction from the unified memory during a first instruction cycle;

(b) determining a type of program instruction;

(c) pre-fetching a next instruction from the unified memory;

(d) saving the pre-fetched instruction in a register when it is determined that the type of program instruction is a first instruction of a loop;

(e) fetching a next instruction from the register when it is determined that the type of program instruction is a last instruction of a loop;

(f) accessing the unified memory with a dummy access during execution of the last instruction of the loop; and

(g) accessing the unified memory, a second time, with a data access during execution of the last instruction of the loop.

[15] 14. (Currently Amended) The method as claimed in claim 14, wherein the method saves the pre-fetched instruction in a stack register when it is determined that the type of program instruction is first instruction of a loop to enable nested loops and interruptible loops, and the method

fetches a next instruction from the stack register when it is determined that the type of program instruction is a last instruction of the loop.